Problem 0. For both programs, I use Python 2.7.11. Also, it is required to install z3 python integration, so "from z3 import *" should works.
You can execute the verifier by executing $ python verifier/verifier.py somefile.java.

Problem 1. For sub-problem 1, I made a enumeration type which name is content, consists of three options: apple, orange, and mixed. Then, I made three variables to denote each crate: realApple, realOrange and realMixed. Also, I added some constraints: i) three variables should be distinct, ii) each crate has content that does not equal to its name, and iii) realMixed crates should has apple or mixed. The execution gave: realApple=orange, realOrange=mixed, realMixed=apple.
For sub-problem 2, I made six variables to denote two models that used in sub-problem 1: realApple1, realApple2, realOrange1, realOrange2, realMixed1 and realMixed2. Then, I added the same constraints to each model, and added conditions that two models are different. The execution gave unsat, which means the solution is unique.
For sub-problem 3, I exactly used same script with sub-problem 2, except realOrange1 and realOrange2 should have apple or mixed. The execution gave: sat, which means two solutions were possible, and the solutions were: realApple1=orange, realOrange1=mixed, realMixed1=apple, and realApple2=mixed, realOrange2=apple, realMixed2=orange.

Problem 2. My program works in four steps: i)run soot.sh, ii) parse, iii) analyze and make assertions and iv) run z3.
(1) Run soot.sh: since there is some problem about soot.sh execution, the program first copies the input java file to soot/ directory, and executes soot to produce shimple file.
(2) Parse: the program eliminates all empty lines and comments, and then finds the testMe method section, and parse the section. The class STMT is used to denote parsed instruction. During parsing, all assignments except int value, Phi equation and assertionDisabled object are ignored.
(3) Analyze: the program executes the STMT objects one by one. Since the program determines the validity of assertion, if the program meets return statement, it returns nothing. However, if the program meets throw statement, it returns current condition. The current condition is made when the program meets if statements. For if statements, the analyze method does recursive call for two cases: taken and not-taken. Current conditions for taken case is (if condition $\wedge$ current condition), and not-taken case is ($\neg$if condition $\wedge$ current condition). Then, it merges two conditions and returns.
(4) Run Z3: the main method calls analyze function to collect all possible conditions to arrive throw statements. Then, it folds them with OR operator to make ultimate one-line assertion to feed z3. If the assertion in the original program is valid, there are no ways to arrive throw statements, so z3 will give unsat. Otherwise, it gives sat, and its satisfying model, so the program simply prints the model results.